Системы тестирования алгоритмов машинного обучения: MLcomp, TunedIt и Полигон*

Лисица А. В., Воронцов К. В., Ивахненко А. А., Инякин А. С., Синцова В. В.
lisitsa@forecsys.ru, voron@ccas.ru, {ivahnenko, inyakin, sintsova}@forecsys.ru
Москва, Вычислительный Центр РАН, ЗАО «Форексис»

«Полигон» — это бесплатная веб-система для тестирования алгоритмов классификации на реальных и модельных данных, разработанная в ВЦ РАН и ЗАО «Форексис». Цель системы — предоставить исследователям и прикладным специалистам удобную площадку для анализа алгоритмов и задач классификации, а также стандартизировать методику тестирования. Аналогичные цели преследуют системы MLcomp и TunedIt. В докладе приводится сравнительный анализ этих трех систем.

Online services for testing machine learning algorithms: MLComp, TunedIt, and Poligon*

Lisitsa A. V., Vorontsov K. V., Ivahnenko A. A., Inyakin A. S., Sintsova V. V.
Dorodnicyn Computing Centre of RAS, Moscow, Russia; Forecsys, Moscow, Russia

Poligon is a free web-based system developed in Dorodnicyn Computing Centre of RAS and Forecsys company for testing machine learning algorithms on real or model data. Poligon provides a convenient platform for the analysis of classification algorithms and tasks as well as standardize the testing procedure. MLcomp and TunedIt have analogous purposes. This paper gives the comparative study of three systems.

Создание новых алгоритмов классификации является сложной задачей. Для того, чтобы алгоритм был востребован, его эффективность должна быть подтверждена экспериментами на реальных данных. Эксперименты должны быть легко воспроизводимыми, чтобы любой ранее полученный результат можно было перепроверить, и стандартизованными, чтобы результаты, полученные разными исследователями, были сопоставимы.

В машинном обучении точной воспроизводимости достичь довольно сложно, так как это требует соблюдения ряда условий.

1. *Идентичность реализации алгоритма.* Авторы алгоритма не всегда предоставляют его реализацию, ограничиваясь описанием идеи. Если же реализация находится в свободном доступе, то трудно гарантировать, что во всех работах используется одна и та же версия. Получение программы или исходных кодов затруднено или невозможно для коммерческих алгоритмов.
2. *Идентичность методики тестирования.* Описания методики кросс-валидации часто поверхностны и не содержат таких важных деталей, как стратификация выборки по классам или по признакам. Более того, точное воспроизведение результатов эксперимента невозможно без знания множества разбиений выборки на обучение и контроль, которое, как правило, генерируется случайным образом и не запоминается.
3. *Идентичность исходных данных.* Хотя данные, как правило, берутся из одного источника (чаще всего из репозитория UCI [4]), детали предварительной обработки (нормировка, заполнение пропусков, и т. д.) тоже часто опускаются.

Если полученные результаты расходятся, то причиной может быть различие и в реализации алгоритмов, и в методике тестирования, и в данных.

Попытки создания систем для автоматизации экспериментов в машинном обучении неоднократно предпринимались в предыдущие десятилетия, но наталкивались на технологические и организационные трудности. Развитие web-технологий привело к созданию в последние годы нескольких общедоступных систем:

— `poligon.MachineLearning.ru` в России (2008);
— `www.TunedIt.org` в Польше (2008);
— `www.MLComp.org` в США (2009).

Поскольку цели этих систем схожи, их функциональные возможности также схожи. Каждая система имеет пополняемый репозиторий задач и алгоритмов. Пользователь системы может ограничивать доступ к добавленной им задаче или алгоритму. Каждая система использует для тестирования алгоритмов методику кросс-валидации, основанную на многократном разбиении исходной выборки данных на «обучение» и «контроль», и позволяет вычислять среднюю ошибку на контроле для любой пары ⟨задача, алгоритм⟩. Каждая система имеет веб-интерфейс для отображения результатов тестирования алгоритмов.

Все системы одинаково определяют круг своих пользователей. Во-первых, это разработчики алгоритмов, которые получают возможность сравни-

Работа выполнена при финансовой поддержке РФФИ (проекты № 10-07-00673, № 08-07-00422) и программы ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

вать свои алгоритмы со стандартными на представительном наборе реальных задач. Во-вторых, это прикладные специалисты, которые получают возможность выбирать алгоритмы, наиболее подходящие для конкретной задачи, особо не вникая в область машинного обучения В-третьих, все системы возникли как университетские проекты и активно используются в учебном процессе и для организации соревнований по анализу данных. В частности, система Полигон используется на кафедре «Интеллектуальные системы» ФУПМ МФТИ и в Школе анализа данных[1] Яндекс.

Однако имеются и существенные различия в том, где хранятся и как исполняются алгоритмы, какие допускаются типы алгоритмов, как строится множество разбиений, какие характеристики качества вычисляются, и в каком виде выдаются результаты тестирования.

Система TunedIt

Система TunedIt начала разрабатываться в 2008 году. Ее основатель и идейный вдохновитель — Marcin Wojnarski (все началось с его диссертационного проекта; сейчас проект поддерживается Варшавским университетом, а также Технологическим инкубатором Варшавского университета).

Система состоит из трех блоков — «базы знаний», «репозитория задач и алгоритмов» и «тестировщика» (*TunedTester*).

TunedTester позволяет запускать алгоритмы и выгружать результаты тестирования на сервер системы. Этот компонент работает локально, поэтому пользователь должен загрузить и установить на свой компьютер программу с сайта www.tunedit.org. Все необходимые для тестировании задачи и алгоритмы загружаются на компьютер пользователя автоматически при запуске тестирования.

Алгоритмы, используемые *TunedTester*'ом должны быть написаны на языке JAVA. Компонент работает с алгоритмами из библиотек WEKA[2], Rseslib[3] и Debellor[4]. Для добавления алгоритма надо встроить его в одну из этих трех библиотек.

В системе есть стандартная процедура тестирования — разбиение задачи на обучение и контроль (случайное, 70/30). Качество алгоритма оценивается как доля ошибок на контроле. Пользователь может дополнить систему собственной процедурой тестирования (*evaluation procedure*), использующей произвольные разбиения задачи и выдающей на выходе любую другую скалярную величину.

Один запуск *TunedTester*'а заключается в выборе процедуры тестирования, алгоритма и задачи. Результаты тестирования могут быть опубликованы в «базе знаний» на сайте системы. «База знаний» предоставляет интерфейс для сравнения результатов различных алгоритмов. Сравнение возможно только в контексте одной задачи и одной процедуры тестирования. Для каждого алгоритма указывается количество запусков процедуры тестирования, среднее значение и стандартное отклонение данной оценки качества на данной задаче. Для стандартной процедуры тестирования оценкой качества является доля ошибок на контроле.

Подход TunedIt имеет следующие ограничения.

1. Ограничение на язык программирования (JAVA).
2. Отсутствуют механизмы контроля мета-параметров, влияющих на процедуру обучения; например, оценка качества работы SVM может вычисляться при различных ядрах.
3. На выходе процедуры тестирования может быть только скалярная величина, что ограничивает возможности детального анализа.
4. Для тестирования алгоритма необходимо иметь его исполняемый код (*java*-класс), что невозможно в случае коммерческих алгоритмов.

Система MLComp

Система MLComp появилась в конце 2009 года. Её создали аспиранты университета Беркли Jacob Abernethy и Percy Liang. Вычислительные мощности системы расположены в «облаках» Amazon Elastic Compute Cloud[5]. Система предлагает автоматизацию тестирования различных методов машинного обучения: алгоритмов классификации, коллаборативной фильтрации, анализа текста и др.

В MLComp другая схема работы с алгоритмами — алгоритмы исполняются не на стороне пользователя, а на сервере системы. Для тестирования алгоритма пользователь загружает свою программу на сайт и запускает её выполнение (run) на одной из задач. Программа выполняется в среде Linux. Система не накладывает ограничений на язык программирования. Так как алгоритмы выполняются на одном сервере и имеют одинаковые вычислительные ресурсы, в рамках MLComp можно сравнивать скорости работы алгоритмов.

В плане процедур тестирования MLComp значительно уступает TunedIt. Проблема заключается в выбранном подходе к тестированию — формат задачи требует явного указания тестовой и контрольной подвыборок, и эта информация хранится в данных задачи. Все тесты алгоритмов на задаче происходят на одном и том же разбиении, что делает невозможным статистически надежную оценку характеристик качества работы алгоритмов.

[1] http://shad.yandex.ru/

[2] http://www.cs.waikato.ac.nz/ml/weka/

[3] http://rseslib.mimuw.edu.pl/

[4] http://www.debellor.org/

[5] http://aws.amazon.com/ec2/

Подход MLComp имеет следующие ограничения.

1. Только одно разбиение задачи на обучение и контроль, в результате низкая статистическая надежность получаемых оценок.
2. Для тестирования алгоритма необходимо передать разработчикам системы его исполняемый код (программу), что невозможно в случае коммерческих алгоритмов.
3. Ограничение на операционную систему (Linux).

Система Полигон

Система Полигон была анонсирована в 2007 году на конференции ММРО-13 [1] и начала функционировать в 2008 году. Она имеет несколько существенных отличий от предыдущих систем.

Во-первых, это распределённая система. Каждый алгоритм запускается на компьютере пользователя (как правило, автора алгоритма) и автоматически выполняет задания, поступающие от сервера Полигона. Задания формируются исходя из того, какие отчёты запрашивают другие пользователи. Таким образом, каждый пользователь, предоставляющий системе свой алгоритм, добровольно выделяет часть своего вычислительного ресурса, но сохраняет полный контроль над программой и исходным кодом алгоритма. Программа может быть реализована на любом языке, поддерживающем web-сервисы. Благодаря этим архитектурным особенностям к системе можно подключать коммерческие версии алгоритмов.

Во-вторых, в Полигоне существенно глубже проработана методика тестирования [1, 2]. Для каждой задачи хранится и используется только одно представительное множество разбиений, что обеспечивает воспроизводимость и статистическую надёжность результатов. Наряду со стандартной характеристикой — частотой ошибок на контроле — вычисляется большое количество скалярных и графических характеристик для детального анализа качества алгоритмов.

Информация, полученная в ходе тестирования, сохраняется в базе данных системы и в дальнейшем может многократно использоваться для выдачи отчётов пользователям.

Полигон оперирует с четырьмя типами объектов: *задача*, *разбиение*, *алгоритм* и *статистика*.

Репозиторий задач классификации.

Задача классификации Z описывается в Полигоне следующим набором данных:

— матрица $F = (F_{ij}) \in \mathbb{R}^{L\times n}$ значений j-го признака на i-ом объекте;
— вектор $y = (y_i)_{i=1}^{L} \in Y^L$ правильных ответов;
— матрица $C = (C_{uv}) \in \mathbb{R}^{m\times m}$ стоимости потерь при ошибочном соотнесении объекта класса u к классу v ($|Y| = m$);
— вектор информации $I = (I_j)_{j=1}^{n}$ о типах признаков (номинальный, вещественный).

Задачи могут быть загружены в формате ARFF[6], либо во внутреннем формате Полигона — в виде нескольких csv-файлов с данными. Пользователь, добавивший задачу в систему, может управлять правами доступа к этой задаче.

Методика тестирования [1, 2].

Для каждой задачи Z формируется набор разбиений по стандартной схеме кросс-валидации «10×5-fold CV» и девять наборов по десять случайных разбиений с одинаковой длиной обучения (от 10% до 90% от длины выборки с шагом в 10%). В контроль и обучение объекты разных классов попадают в такой же пропорции, как и в исходной задаче. Полученные разбиения сохраняются в базе данных системы. Если задача Z однажды тестировалась в Полигоне, то система всегда будет использовать разбиения, сформированные при первом тестировании задачи. Тем самым гарантируется, что все алгоритмы тестируются на одинаковых исходных данных.

Репозиторий алгоритмов классификации.

Алгоритм классификации A принимает на входе:

— матрицу обучающей выборки $F = (F_{ij}) \in \mathbb{R}^{\ell\times n}$,
— вектор $y = (y_i)_{i=1}^{\ell} \in Y^\ell$ правильных ответов на обучающей выборке,
— матрицу $C = (C_{uv}) \in \mathbb{R}^{m\times m}$ стоимости потерь,
— вектор информации I о типах признаков,
— вектор W мета-параметров алгоритма,
— матрицу тестовой выборки $F' = (F'_{ij}) \in \mathbb{R}^{k\times n}$,

и выдаёт на выходе:

— ответы $y' = (y'_i) \in Y^k$ на тестовой выборке,
— матрицу оценок $P' = (P'_{iv}) \in [0,1]^{k\times m}$ принадлежности i-го объекта тестовой выборки v-му классу задачи.

Мета-параметры W задаются пользователем перед запуском алгоритма на тестирование. Экземпляры алгоритма A с различными значениями мета-параметров W (а также различных версий) считаются различными, и результаты их тестирования сохраняются в отдельных записях. Это позволяет Полигону лучше обеспечивать воспроизводимость по сравнению с TunedIt.

Взаимодействие с алгоритмами происходит при помощи специальной прослойки AlgProxy[7], которую должен реализовать автор алгоритма. AlgProxy обеспечивает обмен данными между алгоритмом A и веб-сервисом Полигона. Модель вза-

[6] http://weka.wikispaces.com/ARFF+(stable+version)

[7] В первых версиях Полигона [1, 2] взаимодействие с алгоритмами строилось на манер TunedIt при помощи локальной программы «Вычисляющий сервер». В дальнейшем была выбрана модель взаимодействия при помощи веб-сервиса, и необходимость в «Вычисляющем сервере» отпала.

имодействия основана на подходе «клиент-сервер», где AlgProxy — это клиент, а Полигон — это сервер. Инициатором взаимодействия выступает AlgProxy, который постоянно опрашивает Полигон «не появились ли задания для алгоритма A?». Если ответ положительный, то AlgProxy получает задание и начинает выполнение алгоритма A. Задание содержит данные задачи Z, параметры алгоритма W и набор разбиений $S = (S_q)$. AlgProxy тестирует алгоритм A на разбиениях S и передает результаты в систему Полигон. В упрощенном виде схема работы AlgProxy выглядит так:

1) запросить новое задание;
2) если задания нет, то через N секунд перейти к первому пункту;
3) если задание есть, то запросить данные задачи классификации;
4) провести тестирование алгоритма;
5) отправить результаты тестирования на сервер Полигона.

В документации[8] приведена пошаговая инструкция реализации AlgProxy для подключения собственного алгоритма.

Оценки качества алгоритма для задачи.

Статистика — это функция, которая принимает на входе:

— данные задачи классификации Z,
— набор разбиений $S = (S_q)$ задачи Z, использованных при тестировании алгоритма,
— результаты работы алгоритма (y'_q) и (P'_q) на каждом q-ом разбиении,

и выдаёт на выходе:

— скалярное или векторное значение, описывающее одну из характеристик качества алгоритма.

В текущей версии Полигона рассчитываются следующие статистики (некоторые из них подробнее обсуждались в [2]):

— частота ошибок на обучении и на контроле, а также средняя переобученность (с доверительными интервалами);
— среднее смещение и средняя вариация (характеристики адекватности и устойчивости модели);
— доля «шумовых» (где алгоритм часто ошибается), «пограничных» и «эталонных» (на которых алгоритм ошибается редко) объектов;
— распределение частоты ошибок и переобученности (показатели устойчивости классификации);
— зависимость переобученности и частоты ошибок от длины обучения;
— карта ошибок — точечный график: по оси X частота ошибок на обучении, по оси Y частота ошибок на контроле, каждая точка на карте соответствует одному разбиению;
— разделение ошибок на смещение и вариацию (анализ качества модели [3, 2]);
— ROC-кривая и площадь под ROC-кривой (анализ соотношения ошибок I и II рода [5]);
— распределение отступов (margin) объектов: по оси X — объекты, по оси Y — средний отступ объекта от границы класса (ещё одно разделение объектов на «шумовые», «пограничные» и «эталонные»).

Большинство статистик вычисляются как по всей выборке, так и отдельно по классам, а также отдельно для обучающей и контрольной выборки (если это осмысленно).

Результаты расчетов могут быть выгружены в «сыром» виде, что позволяет пользователю дополнить имеющиеся статистики собственными.

Пока текущая версия Полигона имеет два существенных ограничения: поддерживается только один тип задач (классификации) и отсутствует интерфейс и документация на английском языке.

Выводы

Благодаря архитектурным решениям, основанным на web-сервисах, и стандартизованной методике тестирования, система Полигон обладает лучшими характеристиками воспроизводимости по сравнению с системами TunedIt и MLComp. Полигон не накладывает ограничений на язык программирования и операционную систему при реализации алгоритмов. Алгоритмы исполняются на локальных машинах, что делает возможным тестирование их коммерческих версий. Применяемая методика тестирования позволяет использовать Полигон не только как средство вычисления и хранения результатов тестирования, но и как инструмент для исследования задач и алгоритмов классификации.

Литература

[1] *Воронцов К. В., Инякин А. С., Лисица А. В.* Система эмпирического измерения качества алгоритмов классификации // Всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. — С. 577–581.

[2] *Воронцов К. В., Ивахненко А. А, Инякин А. С., Лисица А. В., Минаев П. Ю.* «Полигон» — распределённая система для эмпирического анализа задач и алгоритмов классификации // Всеросс. конф. ММРО-14. — М.: МАКС Пресс, 2009. — С. 503–506.

[3] *Domingos P.* A unified bias-variance decomposition and its applications: ICML'17. — 2000. — Pp. 231–238.

[4] *Frank A., Asuncion A.* UCI Machine Learning Repository // University of California, Irvine, School of Information and Computer Sciences, 2010. `www.ics.uci.edu/~mlearn/MLRepository.html`.

[5] *Hand D., Till R.* A simple generalization of area under the ROC curve for multiple class classification problems // Machine Learning, 45. — 2001. — Pp. 171–186.

[8] Документация системы в формате *wiki* расположена по адресу `http://www.MachineLearning.ru/wiki/index.php?title=Полигон_алгоритмов/Документация`